NPS ARCHIVE
1967
ADAMS, D.

THE APPLICABILITY OF SPECIAL PURPOSE COMPUTERS TO FAST FOURIER TRANSFORMS

DAVID HUGH ADAMS

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

THE APPLICABILITY OF SPECIAL PURPOSE COMPUTERS TO FAST FOURIER TRANSFORMS

by

David Hugh Adams
Captain, United States Marine Corps
B.E., Vanderbilt University, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ELECTRONICS

from the

NAVAL POSTGRADUATE SCHOOL
September 1967

NPS ARCHIVE
1967
ADAMS, D.

~~Thesis A2315 c.1~~

ABSTRACT

The Fast Fourier Transform is an algorithm for the computation of Discrete Fourier Transforms in less time than allowed by any other algorithm available. The use of special purpose digital machines to reduce those times even further is of interest for real time spectral analysis. The main principles of Fast Fourier Transforms are presented. The design of a full-parallel eight sample processor is presented as a point of reference for comparison with serial and serial-parallel hybrid machines. Carry-Save Addition is introduced and used as the primary arithmetic logic.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

TABLE OF SYMBOLS AND ABBREVIATIONS

Symbols	Definition
Couplets	The combination of two samples in the first pass of a FFT. Referred to as a sum, or a difference, couplet depending on the method of combining.
CSA	Carry-Save Adders or Carry-Save Addition
DFT	Discrete Fourier Transform
F(n)	The n^{th} coefficient of a DFT
FFT	Fast Fourier Transform
N	The number of samples being processed to obtain the DFT
nsec.	Nano-second; 10^{-9} seconds
quartets	The combination of four samples in the second pass of a FFT
usec.	Micro-second; 10^{-6} seconds
Z(k)	The k^{th} sample of a sampled time signal

1. Introduction

In presenting a procedure that will determine the complex Fourier coefficients of a sampled signal, it is first necessary to introduce the discrete Fourier transform (DFT) and its inverse. The DFT is defined by

$$F(n) = \sum_{k=0}^{N-1} Z(k)\exp(-j2\pi nk/N) \qquad k = 0,1,2,.....N-1 \qquad (1)$$

where F(n) is the n^{th} coefficient of the DFT and Z(k) the k^{th} sample of a N sample signal. It is assumed that the N samples are equally spaced and taken at a frequency that is at least twice the highest frequency component of a bandlimited signal. Using the principles of orthogonality it can be shown that the inverse of the DFT is

$$Z(k) = \frac{1}{N} \sum_{k=0}^{N-1} F(n)\exp(j2\pi nk/N) \qquad n = 0,1,2,.... .N-1 \qquad (2)$$

Since the procedure uses successive reductions of a finite series' length by a factor of two, the number of samples, N, must be an integer power, p, of 2 giving $N = 2^p$. It is evident at this point that p reductions or passes will be necessary to obtain a non-serial solution for a given coefficient.

Partitioning equation (1) at the halfway point, it appears as

$$F(n) = \sum_{k=0}^{N/2-1} Z(k)\exp(-j2\pi nk/N) + \sum_{k=N/2}^{N-1} Z(k)\exp(-j2\pi nk/N) \qquad (3)$$

substituting k = m + N/2, in the second half of the equation yields

$$F(n) = \sum_{k=0}^{N/2-1} Z(k)\exp(-j2\pi nk/N) + \sum_{m=0}^{N/2-1} Z(m+\frac{N}{2})\exp[-j2\pi n(m+\frac{N}{2})/N] \qquad (4)$$

and combining under a single summation

$$F(n) = \sum_{k=0}^{N/2-1} \left[Z(k)\exp(-j2\pi nk/N) + Z(k+\frac{N}{2})\exp[-j2\pi n(k+\frac{N}{2})/N] \right] \qquad (5)$$

The common exponential multiplier may be factored and after reduction of the internal exponential one obtains

$$F(n) = \sum_{k=0}^{N/2-1} [Z(k)+Z(k+N/2)\exp(-j\pi n)]\exp(-j2\pi nk/N) \qquad (6)$$

where

$$\exp(-j\pi n) = \begin{cases} +1 & n \text{ even} \\ -1 & n \text{ odd} \end{cases}$$

giving

$$F(n) = \sum_{k=0}^{N/2-1} [Z(k)+Z(k+N/2)]\exp(-j2\pi nk/N) \qquad n \text{ even} \qquad (7)$$

$$F(n) = \sum_{k=0}^{N/2-1} [Z(k)-Z(k+N/2)]\exp(-j2\pi nk/N) \qquad n \text{ odd} \qquad (8)$$

Initiating a new pass, equations (7) and (8) are partitioned at the half-range point

$$F(n) = \sum_{k=0}^{N/4-1} [Z(k)+Z(k+N/2)]\exp(-j2\pi nk/N) + \sum_{k=N/4}^{N/2-1} [Z(k)+Z(k+N/2)]\exp(-j2\pi nk/N) \qquad n \text{ even} \qquad (9)$$

$$F(n) = \sum_{k=0}^{N/4-1} [Z(k)-Z(k+N/2)]\exp(-j2\pi nk/N)$$

$$+ \sum_{k=N/4}^{N/2-1} [Z(k)-Z(k+N/2)]\exp(-j2\pi nk/N) \quad n \text{ odd} \tag{10}$$

substituting $m = k - N/4$, $k = m + N/4$, in the second summations

$$F(n) = \sum_{k=0}^{N/4-1} [Z(k)+Z(k+N/2)]\exp(-j2\pi nk/N)$$

$$+ \sum_{m=0}^{N/4-1} [Z(m+N/4)+Z(m+3N/2)]\exp[-j2\pi n(m+N/4)/N] \quad n \text{ even} \tag{11}$$

$$F(n) = \sum_{k=0}^{N/4-1} [Z(k)-Z(k+N/2)]\exp(-j2\pi nk/N)$$

$$+ \sum_{m=0}^{N/4-1} [Z(m+N/4)-Z(m+3N/2)]\exp[-j2\pi n(m+N/4)/N] \quad n \text{ odd} \tag{12}$$

Combining under a single summation and extracting the common exponentials gives

$$F(n) = \sum_{k=0}^{N/4-1} \left\{ [Z(k)+Z(k+N/2)]+[Z(k+N/4)+Z(k+3N/4)]\exp(-j\pi n/2) \right\}$$

$$\times \exp(-j2\pi nk/N) \qquad n \text{ even} \tag{13}$$

$$F(n) = \sum_{k=0}^{N/4-1} \left\{ [Z(k)-Z(k+N/2)]+[Z(k+N/4)-Z(k+3N/4)]\exp(-j\pi n/2) \right\}$$

$$\times \exp(-j2\pi nk/N) \qquad n \text{ odd} \tag{14}$$

where

$$\exp(-j\pi n/2) = \begin{cases} +1 & n/2 & \text{even} \\ -1 & n/2 & \text{odd} \\ -j & (n-1)/2 & \text{even} \\ +j & (n-1)/2 & \text{odd} \end{cases}$$

After two passes the coefficients are defined by

$$F(n) = \sum_{k=0}^{N/4-1} \{ [Z(k)+Z(k+N/2)]+[Z(k+N/4)+Z(k+3N/4)] \} \exp(-j2\pi nk/N)$$

$$n/2 \quad \text{even} \qquad (15)$$

$$F(n) = \sum_{k=0}^{N/4-1} \{ [Z(k)+Z(k+N/2)]-[Z(k+N/4)+Z(k+3N/4)] \} \exp(-j2\pi nk/N)$$

$$n/2 \quad \text{odd} \qquad (16)$$

$$F(n) = \sum_{k=0}^{N/4-1} \{ [Z(k)-Z(k+N/2)]-j[Z(k+N/4)-Z(k+3N/4)] \} \exp(-j2\pi nk/N)$$

$$(n-1)/2 \quad \text{even} \qquad (17)$$

$$F(n) = \sum_{k=0}^{N/4-1} \{ [Z(k)-Z(k+N/2)]+j[Z(k+N/4)-Z(k+3N/4)] \} \exp(-j2\pi nk/N)$$

$$(n-1)/2 \quad \text{odd} \qquad (18)$$

General characteristics of each reduction are now apparent. They are:

1. For the u^{th} pass a complex multiplier of the form $\exp(-j2\pi n/2^u)$ is generated.

2. The limits of the summation after the u^{th} pass are k=0 to $k=(N/2^u)-1$.

3. For each reduction beginning with some combination of samples, T(k), the multiplicand of the complex multiplier is $T(k+N/2^u)$.

4. The general form after the u^{th} pass is

$$F(n)= \sum_{k=0}^{N/2^{u-1}} [T(k)+\exp(-j2\pi n/2^u)\ T(k+N/2^u)]\exp(-j2\pi nk/N)$$

5. When $u=p$, $N=2^p$, the limits of the summation are k=0 to k=0; the exponential $\exp(-j2\pi nk/N)$ becomes unity and the non-serial solution of F(n) is explicit.

Now, continuing the procedure for the 3^{rd} pass, u=3:

a. the exponential multiplier is

$$\exp(-j2\pi n/2^3) = \exp(-j\pi n/4);$$

b. the limits of summation are k=0 to $k=(N/2^3)-1=N/8-1$;

c. the sample shift is $N/2^3 = N/8$;

d. the general form after the third pass is

$$F(n)= \sum_{k=0}^{N/8-1} [T(k)+\exp(-j\pi n/4)T(k+N/8)]\exp(-j2\pi nk/N).$$

e. if N=8,

$$F(n) = T(0)+\exp(-j\pi n/4)T(N/8).$$

Recalling equations (15) through (18) the following substitutions are made to simply bookkeeping:

$A(k)=Z(k)+Z(k+N/2)$ $\qquad$ $B(k)=Z(k+N/4)+Z(k+3N/4)$

$C(k)=Z(k)-Z(k+N/2)$ $\qquad$ $D(k)=Z(k+N/4)-Z(k+3N/4)$

Also, in any set of equations having common summations, the limits of the summations will only be shown for the first equation of the set.

Rewriting (15-18) yields

$$F(n) = \sum_{k=0}^{N/4-1} [A(k)+B(k)]\exp(-j2\pi nk/N) \qquad n/2 \text{ even} \qquad (15a)$$

$$F(n) = \sum [A(k)-B(k)]\exp(-j2\pi nk/N) \qquad n/2 \text{ odd} \qquad (16a)$$

$$F(n) = \sum [C(k)-jD(k)]\exp(-j2\pi nk/N) \qquad (n-1)/2 \text{ even} \qquad (17a)$$

$$F(n) = \sum [C(k)+jD(k)]\exp(-j2\pi nk/N) \qquad (n-1)/2 \text{ odd} \qquad (18a)$$

After the third pass, the resulting equations are

$$F(n) = \sum_{k=0}^{N/4-1} \left\{ [A(k)+B(k)]+ \exp(-j\pi n/4)[A(k+N/8)+B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$n/2 \text{ even} \qquad (19)$$

$$F(n) = \sum \left\{ [A(k)-B(k)]+ \exp(-j\pi n/4)[A(k+N/8)-B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$n/2 \text{ odd} \qquad (20)$$

$$F(n) = \sum \left\{ [C(k)-jD(k)]+ \exp(-j\pi n/4)[C(k+N/8)-jD(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$(n-1)/2 \text{ even} \qquad (21)$$

$$F(n) = \sum \left\{ [C(k)+jD(k)]+ \exp(-j\pi n/4)[C(k+N/8)+jD(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$(n-1)/2 \text{ odd} \qquad (22)$$

noting

$$\exp(-j\pi n/4) = \begin{cases} +1 & n/4 & \text{even} \\ -1 & n/4 & \text{odd} \\ +.707(1-j) & (n-1)/4 & \text{even} \\ -.707(1-j) & (n-1)/4 & \text{odd} \\ -j & (n-2)/4 & \text{even} \\ +j & (n-2)/4 & \text{odd} \\ -.707(1+j) & (n-3)/4 & \text{even} \\ +.707(1+j) & (n-3)/4 & \text{odd} \end{cases}$$

Performing the complex multiplication yields

$$F(n) = \sum_{k=0}^{N/8-1} \left\{ [A(k)+B(k)]+[A(k+N/8)+B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$n/4 \quad \text{even} \qquad (23)$$

$$F(n) = \sum \left\{ [A(k)+B(k)]-[A(k+N/8)+B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$n/4 \quad \text{odd} \qquad (24)$$

$$F(n) = \sum \left\{ C(k)+.707[C(k+N/8)-D(k+N/8)] \right\}$$

$$-j \; D(k)+.707[C(k+N/8)+D(k+N/8)] \Big\} \Big\} \exp(-j2\pi nk/N)$$

$$(n-1)/4 \quad \text{even} \qquad (25)$$

$$F(n) = \sum \Big[\left\{ C(k)-.707[C(k+N/8)-D(k+N/8)] \right\}$$

$$-j \left\{ D(k)-.707[C(k+N/8)+D(k+N/8)] \right\} \Big] \exp(-j2\pi nk/N)$$

$$(n-1)/4 \quad \text{odd} \qquad (26)$$

$$F(n) = \sum \left\{ [A(k)-B(k)]-j[A(k+N/8)-B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$(n-2)/4 \quad \text{even} \qquad (27)$$

$$F(n) = \sum \left\{ [A(k)-B(k)]+j[A(k+N/8)-B(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$(n-2)/4 \quad \text{odd} \qquad (28)$$

$$F(n) = \sum \left\{ \left\{ C(k)-.707[C(k+N/8)-D(k+N/8)] \right\} + j \left\{ D(k)-.707[C(k+N/8)+D(k+N/8)] \right\} \right\} \exp(-j2\pi nk/N)$$

$$(n-3)/4 \quad \text{even} \qquad (29)$$

$$F(n) = \sum \left\{ C(k)+.707[C(k+N/8)-D(k+N/8)] \right\} + j \left\{ D(k)+.707[C(k+N/8)+D(k+N/8)] \right\} \exp(-j2\pi nk/N)$$

$$(n-3)/4 \quad \text{odd} \qquad (30)$$

Substitution for the functions A,B,C, and D will produce explicit statements of the transform equations after three passes.

Further arithmetic development is avoided for brevity with the statement of the transform equation sets for the 8 sample (N=8) and 16 sample (N=16) cases in Appendices I and II, respectively. Appendix III, in turn, is an illustrative problem in which the spectral lines of a signal are determined and subsequently the originating signal reproduced via the described transform and its inverse.

The procedure just presented is indeed a very simple example of the Fast Fourier Transform (FFT), an algorithm that allows the computation of a time series DFT with fewer actual arithmetic operations than other algorithms available. An arithmetic operation is defined, here, as a complex multiplication followed by a complex addition.

Noting that the straight forward method of computing the DFT requires N^2 operations, Cooley and Tukey [1] showed that less than $2N\log_2 N$ operations are required when the FFT algorithm is used. They also showed that this algorithm requires no more data storage than the storage required for the initial samples, assuming the initial samples are complex. The particular method demonstrated herein, however, was first shown by Sande [2] and later referred to as "Decimation in Frequency" by Cochran, et al, [3] because of the characteristic divisions of the F(n)'s after each pass.

In Fig. 1 a flow diagram for the equations of Appendix I, the eight sample example, is shown. To be sure, this diagram is not entirely general; the assumption of real inputs avoids continual complex additions and allows absolute segregation of the real and imaginary parts of the computed coefficients. Such an assumption has many real world applications and succeeds in reducing the computation time and circuitry by at least a factor of two.

A flow diagram for the computation of the even numbered coefficients of the 16 sample example (equations of Appendix II) is presented in Fig. 2. It is interesting to note that this is, in fact, the same data flow as shown in Fig. 1, with the exception of the first addition cycle. If the first subscript of each basic

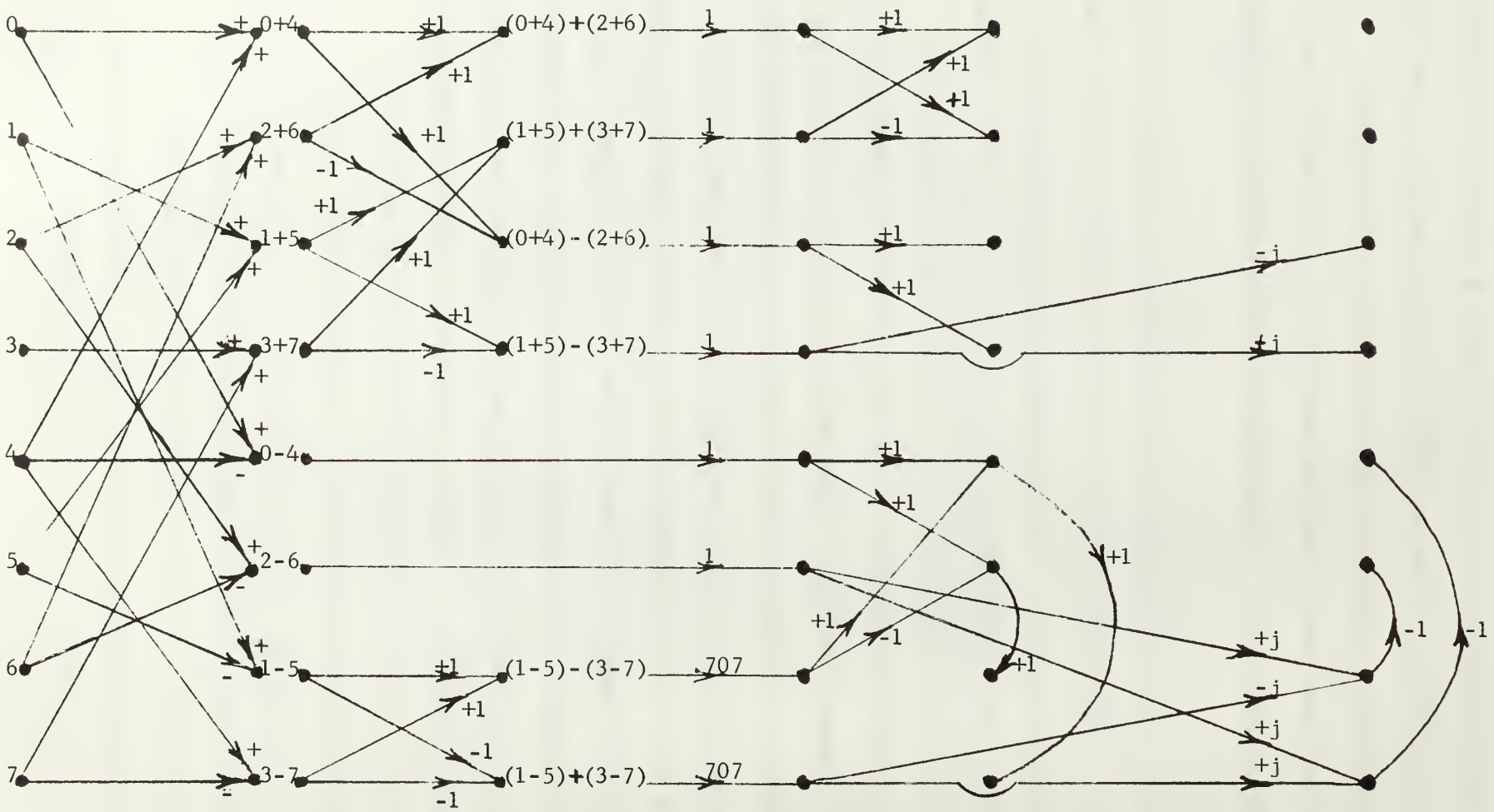


Figure 1. The flow diagram for the coefficients of the eight sample problem. As in all following flow diagrams, the samples are referred to by their subscripts. The value next to an arrow on a flow line references a multiplication, while a node infers a sum.

SAMPLE SUBSCRIPT	FIRST PASS	SECOND PASS	THIRD PASS	F(n) REAL	F(n) IMAG	n
0	0+8	(0+8)+(4+12)=A	A+B			0
1						
2	2+10	(2+10)+(6+14)=B	C+D			8
3						
4	4+12	(1+9)+(5+13)=C	A-B			4
5						
6	6+14	(3+11)+(7+15)=D	C-D			12
7						
8	1+9	(0+8)-(4+12)=E	E			2
9						
10	3+11	(2+10)-(6+14)=F	F			10
11						
12	5+13	(1+9)-(5+13)=G	G-H			6
13						
14	7+15	(3+11)-(7+15)=H	G+H			14
15						

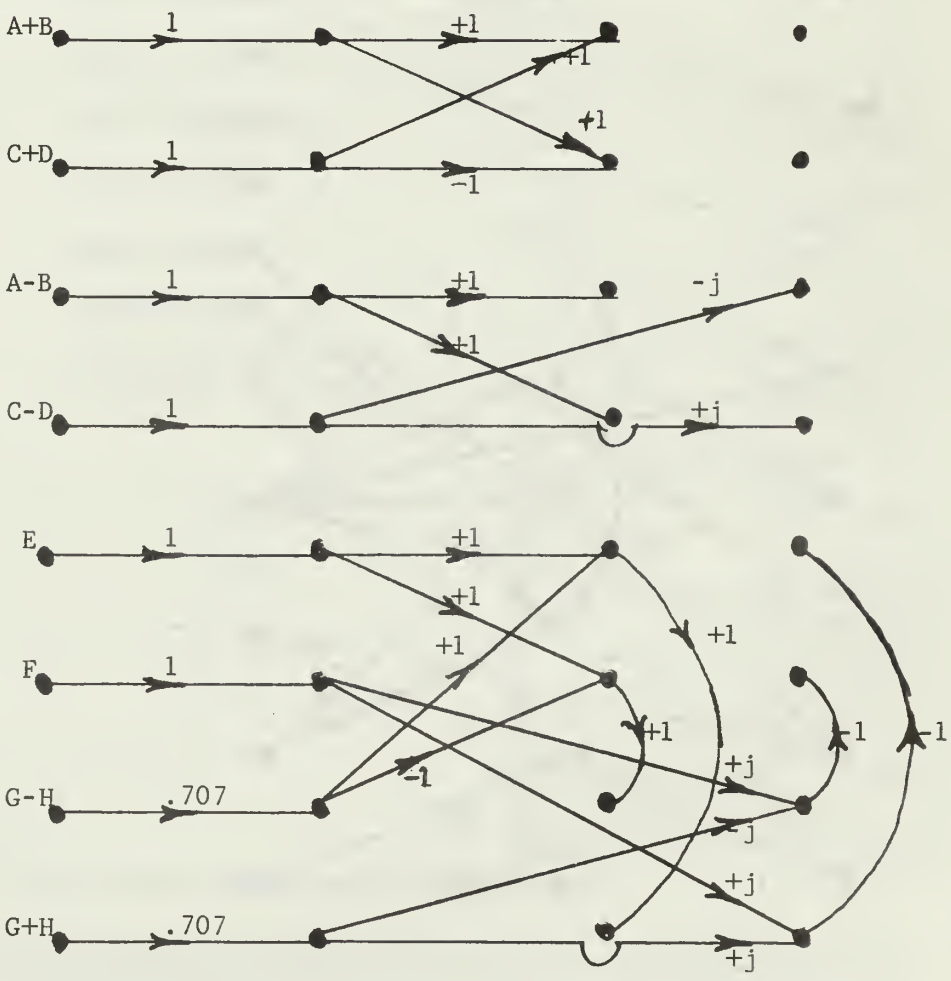


Figure 2. The flow diagram for the even coefficients of the sixteen sample problem.

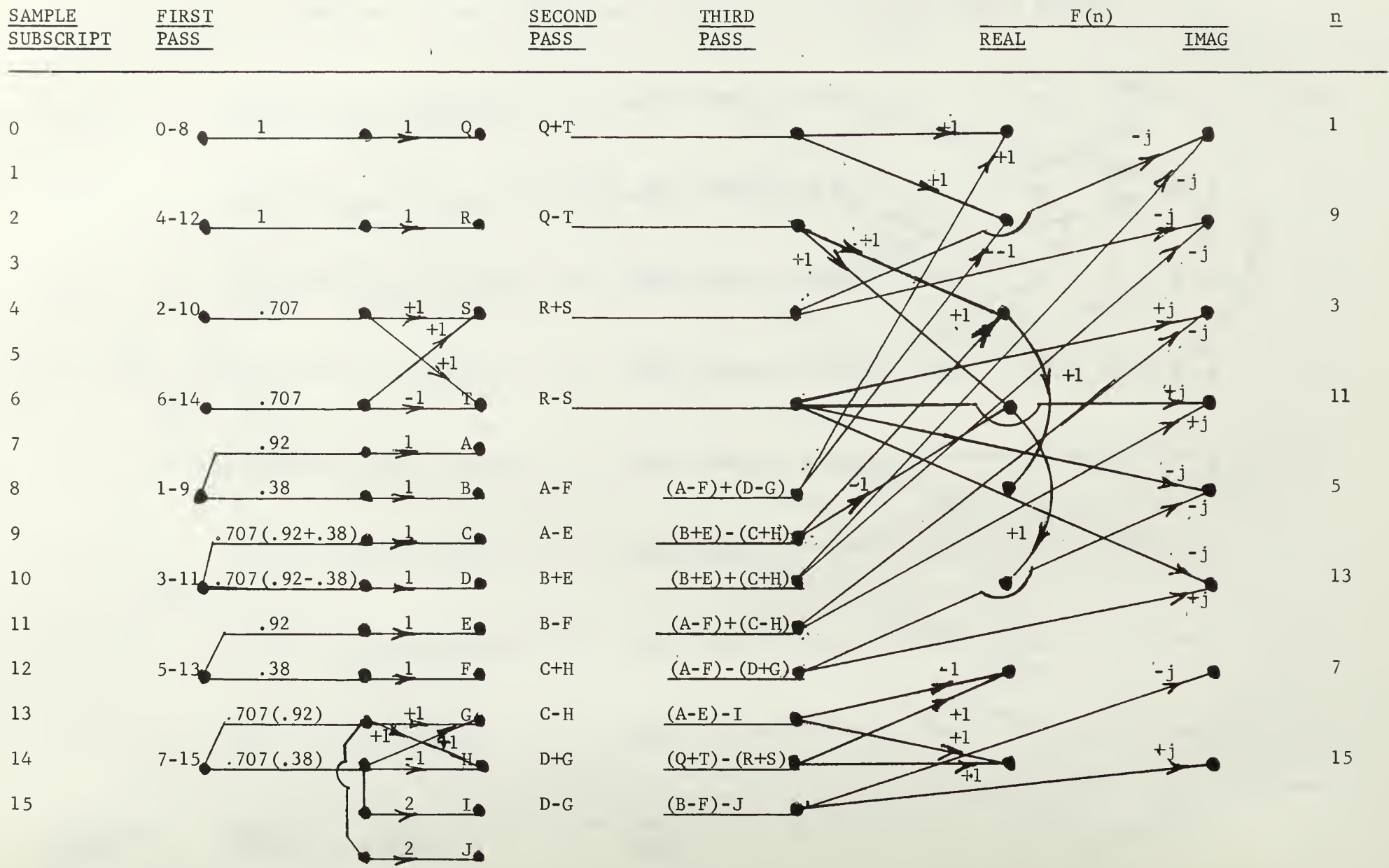


Figure 3. The flow diagram for the odd coefficients of the sixteen sample problem.

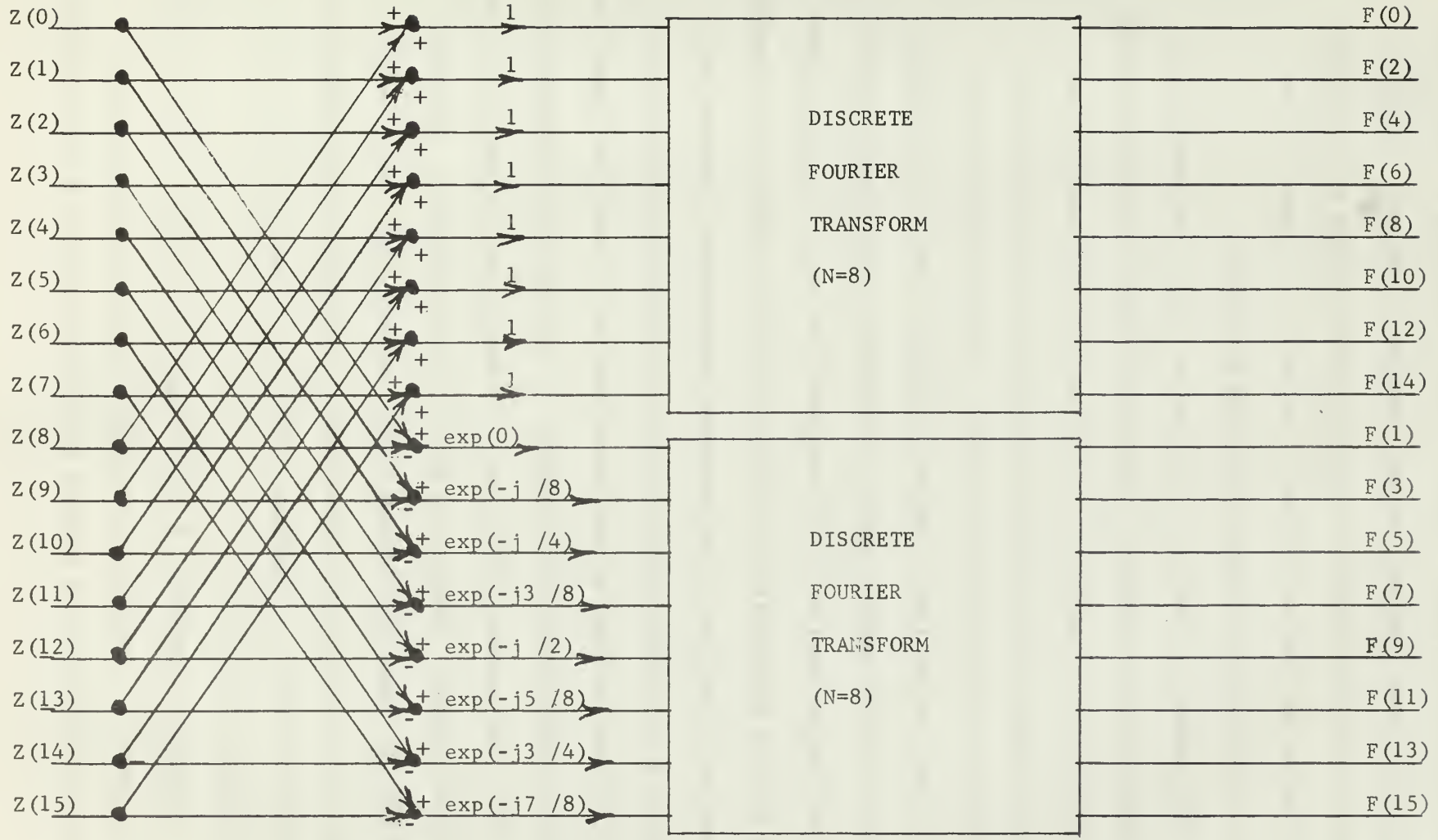


Figure 4. The characteristic reduction of the Fast Fourier Transform by pre-multiplication of the basic couplets with appropriate factors before processing through the next lower order transform.

couplet in the first pass column in Fig. 2 were used to define the couplet, the last statement becomes even more obvious. Figure 3 completes the flow diagram for the 16 sample example by presenting the flow for the odd coefficients. Notably, this type of flow does not take full advantage of the FFT properties.

The pre-multiplication of difference couplets by appropriate exponential factors and then continuing through the next lower order process was shown graphically in [3] for the eight sample case. Figure 4 extends that representation to the 16 sample case and illustrates the ease with which general purpose computers might handle such computations. Comparing Figures 3 and 4, the inherent trade-off between special purpose and general purpose implementations of a particular function is illustrated. The use of recursive procedures and the availability of complex arithmetic makes general purpose computation of the Fig. 4 process most advantageous. Whereas the elimination of complex multiplications and complex input data in the Fig. 3 process produces a more cumbersome set of equations; equations which are, however,more easily adapted to special purpose computation.

It is the purpose of this work to investigate the applicabilities of special purpose computers to Fast Fourier Transforms. The main criterion of comparison between special purpose and general purpose computations will be time versus circuit complexity, where complexity includes the number of components, the size of memory, and the size of the control unit.

A completely parallel, 8 "real" sample, 8 spectral line processor will be used as a vehicle for the "fast" special purpose machine. Extrapolations to greater numbers of sampled inputs and the subsequent

increase in spectral lines will be made with respect to the increased complexity and cost. Further, an investigation of hybrid sequential-parallel machines will be presented.

2. Parallel Processing

The data flow diagrams of the eight sample example is shown in Fig. 1 and the explicit set of equations given in Appendix I.

As the word "parallel" has many connotations when referring to computers, "full parallel" computation implies the simultaneous computation of all equations of the set. This does not, however, mean that the computation of each equation in all passes of the FFT must be independent. In fact, independent computation, when applied to the particular set of equation being examined, would be ridiculous. The computation in each pass is done in parallel.

Examining Fig. 1, six basic combinations of four samples (quartet) and two basic couplets are found at the end of the second pass. Focusing attention on the quartets, for the present, the principle of Carry-Save Adders (CSA) is utilized. As shown in Appendix IV, CSA techniques may be used to obtain exceptionally fast addition of several arguments through logic rather than cyclic operations as in conventional adders. It might be added that CSA may also be used to implement extremely fast multiplication processes. Six 4-argument CSA's will be used at the front of the purposed processor.

At this point, the real parts of F(2) and F(6) and the imaginary part of F(6) are available. The addition of one more rank of CSA's to the imaginary F(6) processor provides the imaginary part of F(2), the negative of the imaginary part of F(6). F(0) and F(4) are obtained by crossfeeding the results of the all even samples and all odd samples summers into two additional two-argument CSA's. The odd-sum must be complemented in one case, producing F(4); the non-complemented case yields F(0).

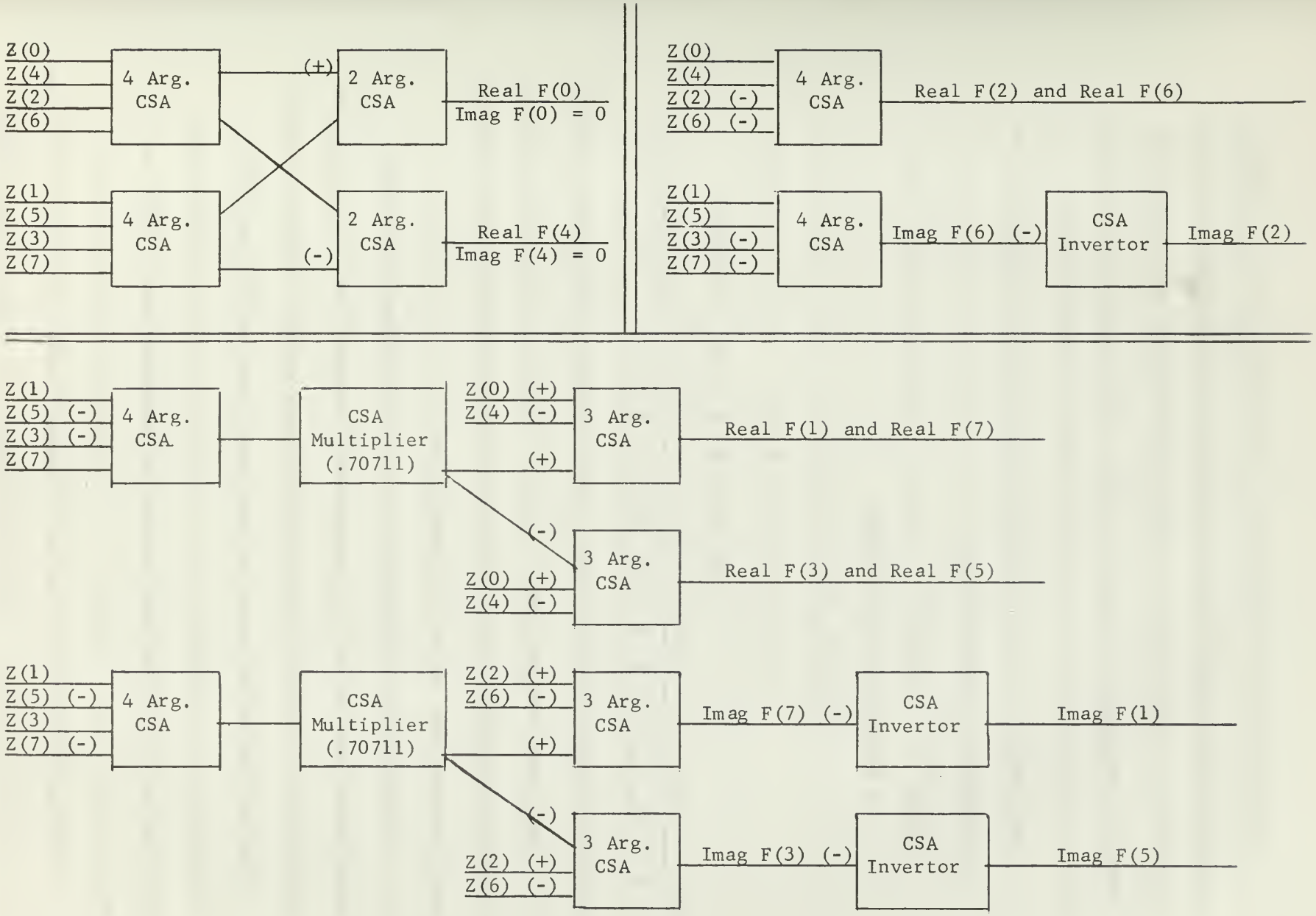


Figure 5. The block diagram of the full-parallel eight sample processor. The numerical signs in parentheses indicate the action within the CSA units. The absence of parentheses indicates a positive input.

To obtain the odd coefficients, the results of the two remaining quartet CSA's must be multiplied by the constant .70711. Multiplication is a shift and add process, dependent on the position of "ones" in the binary multiplier. Since $.70711_{10} = .1011010100_2$ the shift would be to the right and five additions necessitated.

For example, the multiplication of an eight bit quantity M by .70711 would be represented as:

$$
\begin{array}{cccccccc}
M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0 \\
 & M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0 \\
 & & M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0 \\
 & & & M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0 \\
 & & & & M_7 & M_6 & M_5 & M_4 & M_3 & M_2 & M_1 & M_0 \\
\hline
P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0 & P_{-1} & P_{-2} & P_{-3} & P_{-4} & P_{-5} & P_{-6} & P_{-7} & P_{-8}
\end{array}
$$

A CSA multiplier for this example would promulgate the carrys from the -6 digit forward and require a maximum of five arguments (the P_{-1} bit). The maximum number of delays for multiplication by the given constant would then be 5+k for a k bit multiplicand.

Performing the multiplication in two Carry-Save Adder (CSA) multipliers then allows the crossfeeding directly into four 3-argument CSA adders. This gives the respective combination for the real parts of the odd coefficients and the imaginary parts of F(3) and F(7). Complementing the imaginary parts of F(3) and F(7) by an additional CSA rank on each of the sums yields the imaginary part of F(1) and F(5).

Having outlined the complete processor verbally, the block diagram is presented in Fig. 5. Note, the computation of the imaginary parts of F(1) and F(5) are the most involved; therefore, the time analysis is

based on those computations. For the sake of real time estimation, a 12-bit word and the availability of the 20 nsec adders mentioned in Appendix IV are assumed. The number of delays to the final stage is taken from Appendix IV and 11 ripple delays are assumed for the 12-bit word.

The initial four argument processor required three delays, the multiplication four delays, the final three argument addition two delays, and the complement one delay for a total of 10 delays to the final stage. Adding 11 ripple delays gives 21 units of delay for a 420 nsec total time of computation for the eight spectral lines of a signal.

A maximum of k - 1 full adders are required for each bit of a k argument CSA. A measure of complexity, the total number of adders, may be calculated. Thirty-six adders are required for each of the six, 4-argument, 12-bit units for a total of 246 adders for all six units. For each of the two argument summations and the three negation operations, 12 adders are required for a total of 60. For each of the four 3-argument units following the multipliers, 24 adders are required for a total of 96. Finally, for each multiplier 45 adders are needed giving a 90 adder total. The processor then requires a grand total of 492 adders. A completely parallel computation of the eight Fourier coefficients takes less than .5 usec. using five hundred 20 nsec full adders.

Since the initial information may be stored in the same location as the eventual answers, 192 bits of immediate access memory is needed. No control units other than a .5 usec clock and a memory read and write logic are required. If J-K flip-flop memory is used, only one AND gate is needed for control of the memory.

Extension of the methods of computation shown for the eight sample example to a sixteen sample example may be accomplished with one basic assumption; an average of five additions is required for each constant multiplier.

Referring to Fig. 2, the simplest method of obtaining the even coefficients would be to provide eight parallel adders in front of an eight sample processor. For the specified 12-bit word, this requires an added 96 adders to the basic 492 and would add 20 nsec to the computation time.

Computation of the odd coefficients becomes slightly involved. It is found that 1076 adders are required and an approximate computation time of 500 nsecs achieved (based on the multiplication assumption stated above).

Thus a 16-sample processor would require a total of 1664 adders and would produce the 16 complex spectral lines in 500 nsec. It is quite evident that extension of the "full parallel" computation to higher number of imputs becomes extremely expensive in terms of the number of adders required. However, the computation times are extremely fast. It is estimated that almost two million adders would be required for the 1024-sample case. The computations would still be accomplished in less than 2 usec.

It must be remembered that the figures given in this section are for real input values, not complex. To further extend the processors to handle complex input signals would require an approximate 25% increase in adders and a 10% increase in processing time. Thus, the eight and sixteen sample processors would require approximately 600 and 2000 adders respectively for the computation of coefficients in .5 usec.

3. Hybrid Processing

A FORTRAN IV program used to compute the Fast Fourier Transforms on an IBM-360 system was used to give some indication of a fast serial computation of the coefficients for various numbers of sample. The average time to perform an arithmetic operation (complex addition followed by complex multiplication) was estimated at 22 usec. From that average operation time, the maximum computation times for several different numbers of samples were computed and are shown in Fig. 6.

The question of combining the speed of parallel processing with the simplicity of serial operation must be answered with a compromise. It is quite evident that for 16 samples or more, straight parallel computation is unreasonably expensive. With a very low hypothetical price of $10.00 per adder, a 16 sample processor's arithmetic unit alone would cost $20,000.00. Consider, however, a unit which would compute 16 coefficients according to the flow in Fig. 4 by using a single full-parallel eight complex sample processor and serially processing the basic couplet's arithmetic and complex multiplications.

A general 12 x 12 bit CSA multiplier that will give the product in 360 nsec may be constructed with 132 adders and 156 AND gates. With four such multipliers and two simple adders a complex multiplier is constructed that will provide the product of two complex quantities in 380 nsec. Since the basic couplets characteristically fall into sum and difference couplets (see the first pass column in Fig. 2) it seems reasonable to compute both at the same time. By adding a complex subtractor (two words) before the multiplier and a parallel adder, the sum couplet and the difference couplet with complex multiplication are computed in 400 nsec. Such a unit would require approximately 575

adders and 624 AND gates. If 9 AND gates are considered equivalent to one full adder, the above circuit is equivalent to 640 adders.

At this point the "hybrid" processor consist of one eight-complex-sample processor and one arithmetic unit that computes the sum couplet, the difference couplet, and multiplies the difference couplet by a complex constant in a single 400 nsec operation.

For the 16 sample problem of Fig. 4, the arithmetic unit would only be required to cycle through eight calculations (First Pass), then the eight sample unit would cycle twice. Assuming a .25 usec read and write time, one complete arithmetic operation could be completed in less than 1.25 usec. The eight arithmetic operations would require 10 usec and the two cycles of the eight sample unit 9 usec (4.5 usec per cycle) for a 19 usec computation time. The above calculation times have been extended for various numbers of sampled inputs and the results are shown in Fig. 6.

The hybrid processor's arithmetic unit at this point consist of a 1240 adder unit. However, the processor must have a control unit and a control memory. The control memory requires a maximum of N/2 complex multipliers plus the control program. The control unit must be able to index the pass numbers, the arithmetic operations, and the eight sample process pass number. Assuming a single word in memory is equivalent to a full adder in cost and complexity, a possible basis of comparison is achieved. Since two words of memory are needed for each multiplier there will be N words or, equivalently, N adders required in the control memory of a N sample unit. An assumption of a 200 adder equivalent control unit, including the control program memory is a reasonable estimate. For the range of sample inputs from

8 to 1024 this value should remain relatively constant. In general, the control unit then requires the equivalent of N + 200 adders.

The 16 sample hybrid processor is the equivalent of 1450 full adders. Figure 7 illustrates the relative complexities of the hybrid, the parallel, and the serial processor. The basic requirements for sample input and coefficient output memory is the same for all three processes and is not included in the complexity figures.

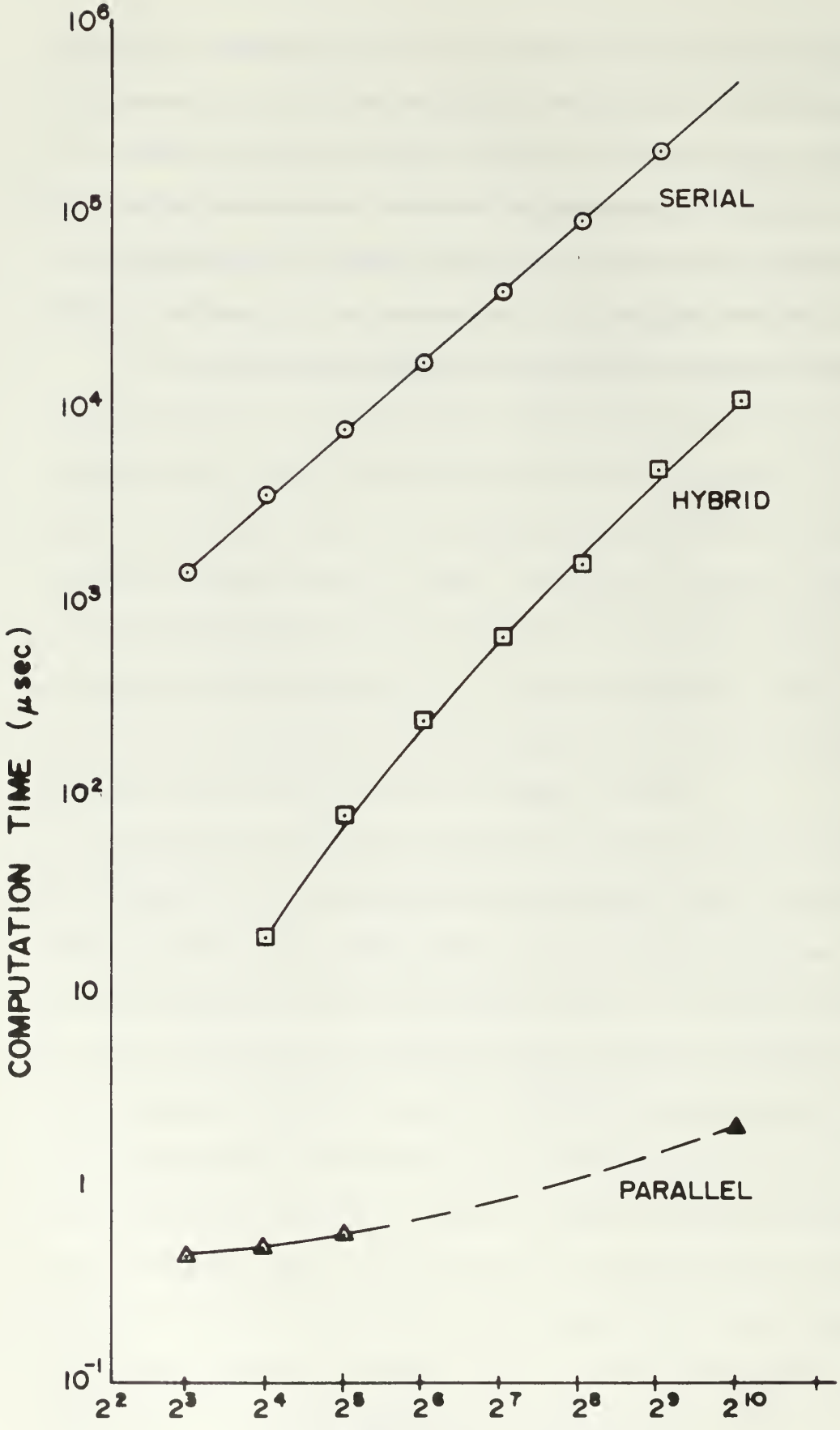


FIG. 6. COMPUTATION TIME Vs. NUMBER OF SAMPLES

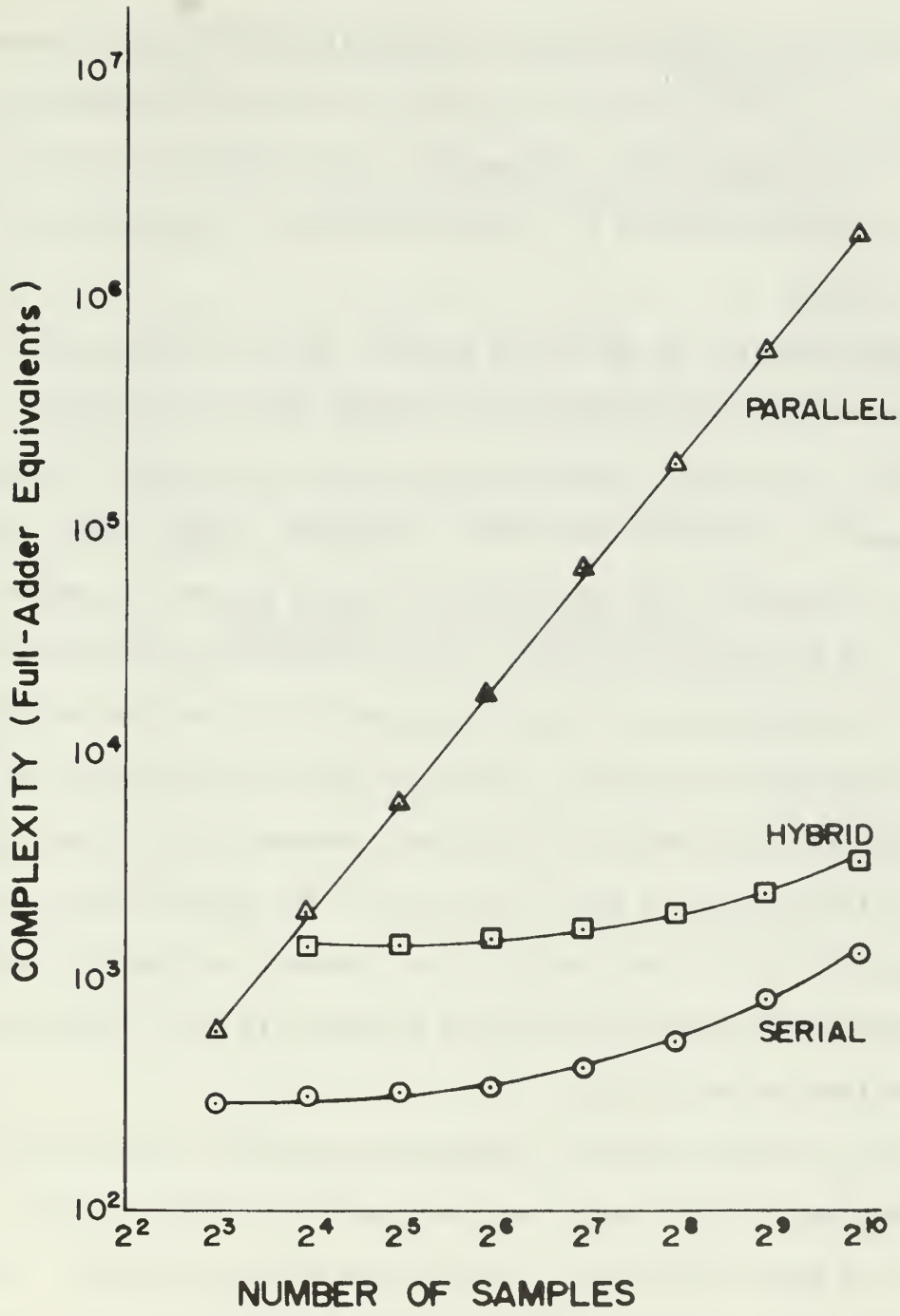


FIGURE 7. COMPLEXITY Vs. NUMBER OF SAMPLES

4. Conclusions

Figure 6 graphically shows the superiority of the parallel processor with respect to speed of computation while Fig. 7 demonstrates the enormity of size and complexity required to obtain those speeds. In reverse, Fig. 7 shows the serial processor with the complexity advantage and being extremely slow in computation, as shown in Fig. 6.

The combining of serial and parallel operations succeeded in incorporating the better qualities of each. Certainly, the complexity of the hybrid system follows the same trend as the serial processor in that it is relatively flat in the log-log plot. This was to be expected. In presenting the hybrid system, the major nature of the system is serial. The parallel characteristics are found in the handling of several computations at one time and reducing substantially the arithmetic operation time by batch processing through the eight sample full-parallel processor in the final stages. On the log-log plot of Fig. 6 the slope of the hybrid curve is slightly greater than that of the serial curve. However, at the last point presented (1024 samples) the hybrid processor is still 500 times faster than the serial unit.

It is evident from the computation times shown that definite applications for the special purpose handling of FFT's do exist. In fact, for many applications, real-time analysis is possible. A central problem remains, that of analog to digital conversion of the data in times that will be able to fully utilize the speed of the processors.

BIBLIOGRAPHY

1. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, Vol. 19, No. 90, (1965), pp. 297-301.

2. W. M. Gentlemen and G. Sande, "Fast Fourier Transforms for Fun and Profit," 1966 Fall Joint Computer Conf., AFIPS Proc., Vol. 29, Washington, D. C.: Spartan, 1966, pp. 563-578.

3. W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. C. Maling, Jr., D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the Fast Fourier Transform?", IEEE Transactions on Audio and Electroacoustics, Vol. AU-15, No. 2, June 1967, pp. 45-55.

4. M. S. Schmookler, "Microelectronics Opens the Gate to Faster Digital Computers," Electronic Design, Ed. 16, July 5, 1966, pp. 52-57.

5. Flores, I., The Logic of Computer Arithmetic, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1963.

APPENDIX I

EQUATIONS FOR EIGHT SAMPLE APPLICATION

The following set of equations defines the DFT for n=8. The equations are expressed in terms of the sample number (sample subscript) rather than using the subscripted notation of the Introduction. In this notation the couplet [Z(0)+Z(4)] would appear as (0+4), and non-integer numbers are multiplicative constants.

$$F(0) = [(0+4)+(2+6)] + [(1+5)+(3+7)]$$

$$F(1) = \{(0-4)+.707[(1-5)-(3-7)]\} - j\{(2-6)+.707[(1-5)+(3-7)]\}$$

$$F(2) = [(0+4)-(2+6)]-j[(1+5)-(3+7)]$$

$$F(3) = \{(0-4)-.707[(1-5)-(3-7)]\} + j\{(2-6)-.707[(1-5)+(3-7)]\}$$

$$F(4) = [(0+4)+(2+6)]-[(1+5)+(3+7)]$$

$$F(5) = \{(0-4)-.707[(1-5)-(3-7)]\} - j\{(2-6)-.707[(1-5)+(3-7)]\}$$

$$F(6) = [(0+4)-(2+6)]+j[(1+5)-(3+7)]$$

$$F(7) = \{(0-4)+.707[(1-5)-(3-7)]\} + j\{(2-6)+.707[(1-5)+(3-7)]\}$$

APPENDIX II

EQUATIONS FOR SIXTEEN SAMPLE APPLICATION

The set of equations beginning on the following page defines the DFT for N=16. As in Appendix, I, the equations are expressed in terms of the sample number.

$$F(0) = [(0+8)+(4+12)]+[(2+10)+(6+14)] + [(1+9)+(5+13)]+[(3+11)+(7+15)]$$

$$F(1) = \{[(0-8)+.707[(2-10)-(6-14)]] +.92 [(1-9)+.707[(3-11)-(7-15)]] -.38 [(5-13)+.707[(3-11)+(7-15)]]\}$$
$$-j\{[(4-12)+.707[(2-10)+(6-14)]] +.38 [(1-9)+.707[(3-11)-(7-15)]] +.92 [(5-13)+.707[(3-11)+(7-15)]]\}$$

$$F(2) = \{[(0+8)-(4+12)]+.707 [(1+9)-(5+13)]-[(3+11)-(7+15)]\}$$
$$-j\{[(2+10)-(6+14)+.707 [(1+9)-(5+13)]+[(3+11)-(7+15)]\}$$

$$F(3) = \{[(0-8)-.707[(2-10)-(6-14)] +.38 (1-9)-.707[(3-11)-(7-15)]] +.92 (5-13)-.707[(3-11)+(7-15)]\}$$
$$+j\{[(4-12)-.707[(2-10)+(6-14)]] -.92 [(1-9)-.707[(3-11)-(7-15)] +.38 [(5-13)-.707[(3-11)+(7-15)]]\}$$

$$F(4) = [[(0+8)+(4+12)]-[(2+10)+(6+14)]] -j [[(1+9)+(5+13)]-[(3+11)+(7+15)]]$$

$$F(5) = \{[(0-8)-.707[(2-10)-(6-14)]] -.38 [(1-9)-.707[(3-11)-(7-15)]] -.92 [(5-13)-.707[(3-11)+(7-15)]]\}$$
$$-j\{[(4-12)-.707[(2-10)+(6-14)]] + .92 [(1-9)-.707[(3-11)-(7-15)]] -.38 [(5-13)-.707[(3-11)+(7-15)]]\}$$

$$F(6) = \{[(0+8)-(4+12)]-.707 [[(1+9)-(5+13)]-[(3+11)-(7+15)]]\}$$
$$+j\{[(2+10)-(6+14)]-.707 [[(1+9)-(5+13)]+[(3+11)-(7+15)]]\}$$

$$F(7) = \{[(0-8)+.707[(2-10)-(6-14)]] - [(4-12)+.707[(2-10)+(6-14)]] -.92 [[(1-9)+.707[(3-11)-(7-15)]]$$
$$- [(5-13)+.707[(3-11)+(7-15)]]]\}$$
$$-j(.38) [[(1-9)+.707[(3-11)-(7-15)]] - [(5-13)+.707[(3-11)+(7-15)]]]$$

$$F(8) = [[(0+8)+(4+12)]+(2+10)+(6+14)]] - [[(1+9)+(5+13)]+[(3+11)+(7+15)]]$$

$$F(9) = \{[(0-8)+.707[(2-10)-(6-14)] -.92 [(1-9)+.707[(3-11)-(7-15)]] +.38 [(5-13)+.707[(3-11)+(7-15)]]\}$$
$$-j\{[(4-12)+.707[(2-10)+(6-14)]] -.38 [(1-9)+.707[(3-11)-(7-15)]] -.92 [(5-13)+.707[(3-11)+(7-15)]]\}$$

$$F(10) = [(0+8)-(4+12)]-.707\,[(1+9)-(5+13)]-[(3+11)-(7+15)]$$
$$-j[(2+10)-(6+14)]-.707\,[(1+9)-(5+13)]+[(3+11)-(7+15)]$$

$$F(11) = (0-8)-.707[(2-10)-(6-14)] - .38\,(1-9)-.707[(3-11)-(7-15)] - .92\,(5-13)-.707[(3-11)+(7-15)]$$
$$+j[(4-12)-.707[(2-10)+(6-14)] + .92\,(1-9)-.707[(3-11)-(7-15)] - .38\,(5-13)-.707[(3-11)+(7-15)]$$

$$F(12) = [(0+8)+(4+12)]-[(2+10)+(6+14)] + j\,[(1+9)+(5+13)]-[(3+11)+(7+15)]$$

$$F(13) = (0-8)-.707[(2-10)-(6-14)] + .38\,(1-9)-.707[(3-11)-(7-15)] + .92\,(5-13)-.707[(3-11)+(7-15)]$$
$$-j[(4-12)-.707[(2-10)+(6-14)] - .92\,(1-9)-.707[(3-11)-(7-15)] + .38\,(5-13)-.707[(3-11)+(7-15)]$$

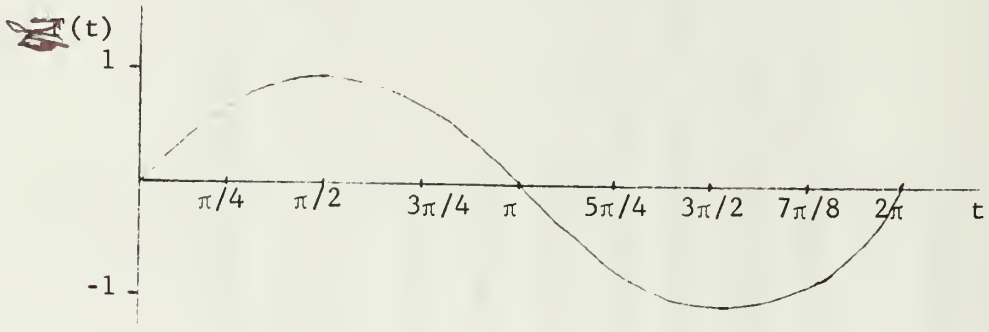
$$F(14) = [(0+8)-(4+12)]+.707\,[(1+9)-(5+13)]-[(3+11)-(7+15)]$$
$$+j[(2+10)-(6+14)]+.707\,[(1+9)-(5+13)]+[(3+11)-(7+15)]$$

$$F(15) = (0-8)+.707[(2-10)-(6-14)] - (4-12)+.707[(2-10)+(6-14)] - .92\,(1-9)+.707[(3-11)-(7-15)]$$
$$- (5-13)+.707[(3-11)+(7-15)]$$
$$+j(.38)\,(1-9)+.707[(3-11)-(7-15)] - (5-13)+.707[(3-11)+(7-15)]$$

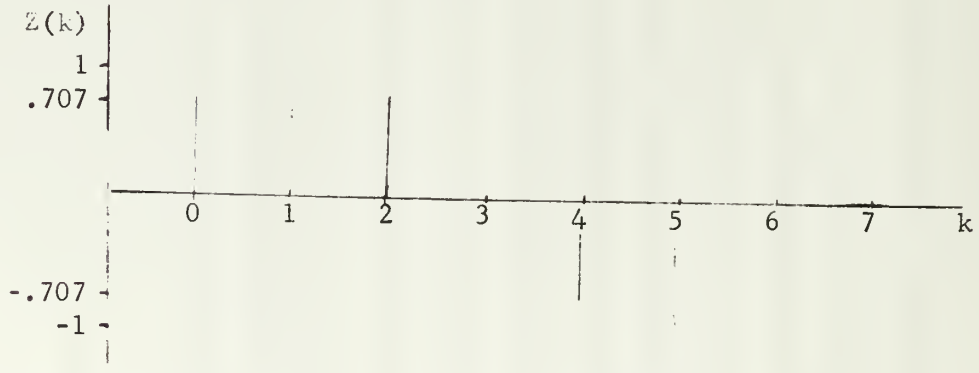
APPENDIX III

ILLUSTRATIVE EXAMPLE OF THE DISCRETE FOURIER TRANSFORM AND ITS INVERSE

A time signal, $f(t)$, is sampled eight times at equally spaced intervals. For convenience, the signal is a sinewave and the interval $\pi/4$ is chosen.



The sampled signal appears as



$Z(0)=Z(2)=.707$ $\quad$ $Z(1)=1$
$Z(4)=Z(6)=-.707$ $\quad$ $Z(5)=-1$
$Z(3)=Z(7)=0$

Forming the basic couplets of the equations in Appendix I.

$Z(0)+Z(4)=0$ $\quad$ $Z(0)-Z(4)=1.414$
$Z(2)+Z(6)=0$ $\quad$ $Z(2)-Z(6)=1.414$
$Z(1)+Z(5)=0$ $\quad$ $Z(1)-Z(5)=2$
$Z(3)+Z(7)=0$ $\quad$ $Z(3)-Z(7)=0$

Now, solving for the coefficients

$F(0)=(0+0)+(0+0) = 0$

$F(1)=[1.414+.707(2-0)]-j[1.414+.707(2+0)] = 2.828(1-j)$

$F(2)=(0-0)-j(0-0) = 0$

$F(3)=[1.414-.707(2-0)]+j[1.414-.707(2+0)] = 0+j0 = 0$

$F(4)=(0+0)-(0+0) = 0$

$F(5)=[1.414-.707(2-0)]-j[1.414-.707(2+0)] = 0-j0 = 0$

$F(6)=(0-0)+j(0+0) = 0$

$F(7)=[1.41+.707(2-0)]+j[1.414+.707(2+0)] = 2.828(1+j)$

If the foregoing procedure is correct, application of the inverse transform should produce the original sample values. Noting that the only non-zero coefficients are F(1) and F(7), the inverse is written

$Z(k)=1/8\ F(1)\exp(j\pi k/4)+F(7)\exp(j\pi k/4)$

and the Z(k)'s are

$Z(0)=(2.828/8)[(1-j)+(1+j)] = .707$

$Z(1)=(2.828/8)[(1-j)(.707)(1+j)+(1+j)(.707)(1-j)] = 1$

$Z(2)=(2.828/8)[(1-j)(j)+(1+j)(-j)] = .707$

$Z(3)=(2.828/8)[(-.707)(1-j)^2+(-.707)(1+j)^2] = 0$

$Z(4)=(2.828/8)[(1-j)(-1)+(1+j)(-1)] = -.707$

$Z(5)=(2.828/8)[(-.707)(1+j)(1-j)+(-.707)(1-j)(1+j)] = -1$

$Z(6)=(2.828/8)[(1-j)(-j)+(1+j)(j)] = -.707$

$Z(7)=(2.828/8)[(.707)(1-j)^2+(.707)(1+j)^2] = 0$

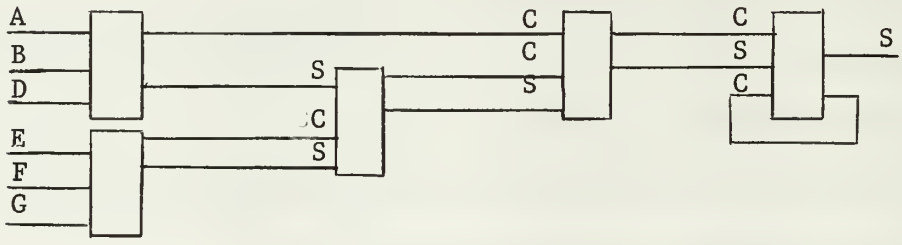
The above Z(k) match precisely the original data; thus showing the correctness of the development

APPENDIX IV

CARRY-SAVE ADDERS

Basically a full adder has three inputs (two addends and a carry-in) and two outputs (sum and carry-out). Thus, it may be looked upon as a reduction process of from three to two arguments. Considering the "carry-in" as a third addend, it is apparent that the addition of more than two quantities may be handled by cascading the afore mentioned reduction process. Such implementation of full adders has led to both the process and the adders themselves being called Carry-Save Adders (CSA).

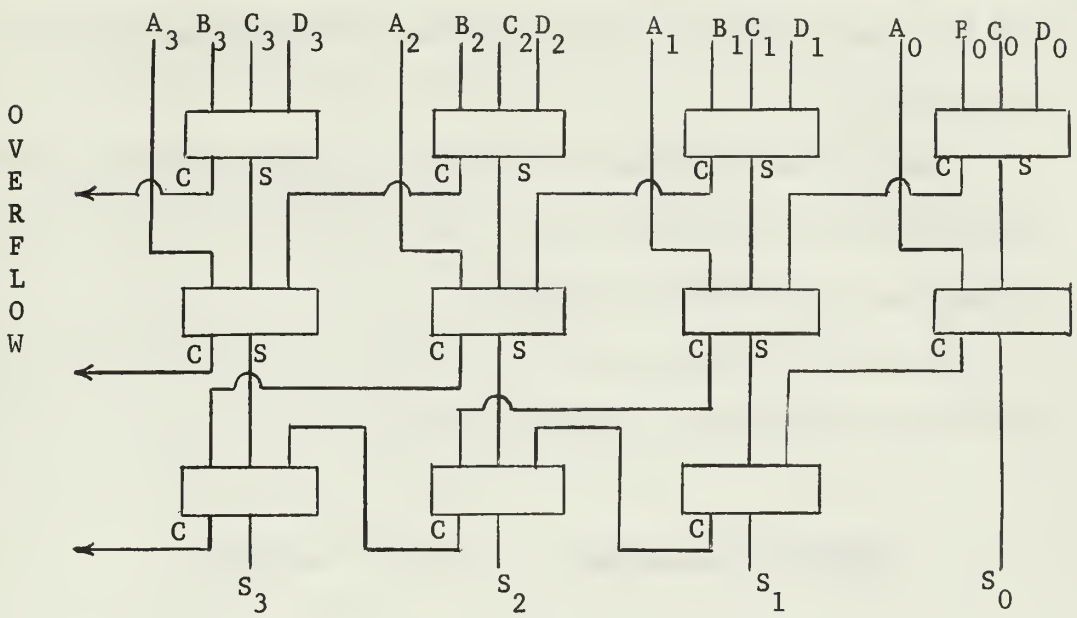
If more than three addends are being summed more than one initial CSA would be required. The figure below schematically shows the reduction process for a six argument addition.



Schematic representation of a six argument Carry-Save Addition

Note that this is strictly a representation of the reduction process. In the actual process the "carry-out" of each adder must go to the next higher significant bit train. Its position, as shown in the diagram, would be taken by the carries from the next lower significant bit train. It is noted here that the CSA process generally requires k-1 units for the reduction of k addends. The final stage (level) of the CSA process is of interest since the propagation of the carries are of a "ripple" nature.

The following diagram presents a complete four-bit adder for four addends (A,B,C, & D). It is obvious that the CSA process



Block diagram of a four-bit, four addend,
Carry-Save Adder (an addition of four 4-bit words)

is strictly a logic function. With the assumption of a realistic signal delay time through each logic level, the total computation time may be calculated. Although pressing the state of the art at this time, an assumption of a 20 nsec full adder is not unrealistic. Thus, for the given adder, three levels of delay plus the final carries' ripple through two delay levels gives a total of five delays, or a 100 nsec addition of four, 4-bit arguments.

With increasing word lengths the computation time increases linearly for the CSA process. This characteristic is the result of adding one more stage of carry ripple for each additional bit. For a 12-bit version of the four argument adder, there would be 13 units of delay and an addition time of 260 nsec.

The use of carry look ahead techniques can reduce the ripple time by a factor of two per stage of delay. For the 12-bit example, the three initial units of delay are summed with 10 half-units for 8 delays and a 160 nsec addition of four, 12-bit, quantities is possible.

Finally, ones complement arithmetic is appropriate since it is easily obtained by using the two's complement and just adding one in the least significant bit train.

The following table gives the number of delays as a function of the number of arguments in a CSA process.

Number of Arguments	Number of Delays
2	1
3	2
4	3
5,6	4
7-9	5
10-13	6
14-19	7
20-28	8

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Commandant of the Marine Corps (Code A03C) Headquarters, U. S. Marine Corps Washington, D. C. 22214	1
4.	Professor Harold A. Titus Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	10
5.	Mr. Art Bittel Naval Weapons Center Code 3040 China Lake, California	2
6.	Capt. David H. Adams, USMC 1039 Halsey Drive Monterey, California 93940	3

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
The Applicability of Special Purpose Computers to Fast Fourier Transforms

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Masters Thesis - September 1967

5. AUTHOR(S) *(Last name, first name, initial)*
Adams, David H., Captain, U. S. Marine Corps

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
September 1967	45	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. AVAILABILITY/LIMITATION NOTICES
This document is subject to special export controls and each transmittal to foreign government or foreign nationals may be made only with prior approval of the Naval Postgraduate School.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY
a) U. S. Marine Corps
b) Naval Weapons Center
China Lake, California

13. ABSTRACT

The Fast Fourier Transform is an algorithm for the computation of Discrete Fourier Transforms in less time than allowed by any other algorithm available. The use of special purpose digital machines to reduce those times even further is of interest for real time spectral analysis. The main principles of Fast Fourier Transforms are presented. The design of a full-parallel eight sample processor is presented as a point of reference for comparison with serial and serial-parallel hybrid machines. Carry-Save Addition is introduced and used as the primary arithmetic logic.

DD FORM 1 JAN 64 1473

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Fast Fourier Transform						
Discrete Fourier Transform						
Carry-Save Adders						
Real Time Computation						

thesA2315
The applicability of special purpose com
3 2768 001 90913 8
DUDLEY KNOX LIBRARY